OIPE IAP12
OCT 0 2 2006
PATENT & TRADEMARK OFFICE

AF/IFW

TRANSMITTAL OF APPEAL BRIEF (Large Entity)

Docket No.
ITL.0663US

In Re Application Of: **DAVID K. POULSEN, ET AL.**

Application No.	Filing Date	Examiner	Customer No.	Group Art Unit	Confirmation No.
10/039,789	January 2, 2002	Michael J. Yigdall	21906	2192	9218

Invention: **PROVIDING PARALLEL COMPUTING REDUCTION OPERATIONS**

COMMISSIONER FOR PATENTS:

Transmitted herewith is the Appeal Brief in this application, with respect to the Notice of Appeal filed on:
July 24, 2006

The fee for filing this Appeal Brief is: **$500.00**

☒ A check in the amount of the fee is enclosed.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 20-1504 I have enclosed a duplicate copy of this sheet.

☐ Payment by credit card. Form PTO-2038 is attached.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

[illegible]
Signature

Dated: **September 29, 2006**

E.E. "Jack" Richards, II, Reg. No. 53,514
Trop, Pruner & Hu, P.C.
1616 S. Voss Rd., Ste. 750
Houston, Texas 77057
(512) 418-9944
(713) 468-8883 [Fax]

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" [37 CFR 1.8(a)] on

September 29, 2006
(Date)

Rose Marie Henderson
Signature of Person Mailing Correspondence

ROSE MARIE HENDERSON
Typed or Printed Name of Person Mailing Correspondence

cc:

P30LARGE/REV07

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	DAVID K. POULSEN, ET AL.	§	Group Art Unit:	2192
		§		
		§		
Serial No.:	10/039,789	§		
		§	Examiner:	Michael J. Yigdall
Filed:	January 2, 2002	§		
		§		
For:	PROVIDING PARALLEL COMPUTING REDUCTION OPERATIONS	§	Atty. Dkt. No.:	ITL.0663US (P12629)

Mail Stop **Appeal Brief-Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

10/03/2006 CNGUYEN 00000021 10039789
01 FC:1402 500.00 OP

Date of Deposit: Sept. 29, 2006
I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Mail Stop Appeal Brief Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Rose Marie Henderson

Rose Marie Henderson

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is Intel Corporation, the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

There are no related interferences or appeals.

III. STATUS OF CLAIMS

Claims 1, 3, 6-8, 10-15, 18, 20-26 are rejected. Each rejection is appealed.

IV. STATUS OF AMENDMENTS

No amendments are presently pending.

V. SUMMARY OF CLAIMED SUBJECT MATTER

At this point, no issue has been raised that would suggest that the words in the claims have any meaning other than their ordinary meanings. Nothing in this section should be taken as an indication that any claim term has a meaning other than its ordinary meaning.

Claims 1, 8, and 15 are independent claims. Support for the subject matter of these claims can be found, for example, in the Specification as filed at p.5, ln. 21 – p. 6, ln. 25; p. 6, ln. 26 – p. 7, ln. 11; p. 7, ln. 12 – p. 10, ln. 18; and FIGS. 2 – 5, 8, 10 and the corresponding discussion concerning the figures found in the Specification, as described further below. The following is a further concise explanation of the subject matter defined in the independent claims of the present Appeal, also identified by page and line number in the Specification.

Referring to Figure 2, a first code 202 may be a source program that may be written in a programming language. A few examples of programming languages are Fortran 90, Fortran 95 and C++. The first code 202 may be a source program that may have been converted to parallel form by annotating a corresponding sequential computer programming with directives according to a parallelism specification such as OpenMP. In other embodiments, the first code 202 may be coded in parallel form in the first instance. Specification, p. 5, ln. 21 – p.6, ln. 2.

These annotations may designate, parallel regions of execution that may be executed by one or more threads, single regions that may be executed by a single thread, and instructions on how various program variables should be treated in the parallel and single regions. The parallelism specification in some embodiments, may include a set of directives such as the directive "!$omp reduce" which will be explained in more detail below. Specification, p. 6, lns. 3 – 10.

In some embodiments, parallel regions may execute on different threads that run on different physical processors in the parallel computer system, with one thread per processor. However, in other embodiments, multiple threads may execute on a single processor.

In some embodiments, the first code 202 may be an annotated source code and may be read into a code translator 28. Translator 28 may perform a source-code-to-source-code level transformation of OpenMP parallelization directives in the first code 202 to generate, in some embodiments, Fortran 95 source code in the second code 204. However, as previously mentioned, other programming languages may be utilized. In addition, the translator 28 may perform a source-to-assembly code level or a source-to-intermediate level transformation of the first code 202. Specification, p. 6, lns. 16 – 25.

The compiler 26 may receive the second code 204 and may generate an object code 210. In an embodiment, the compilation of translated first code 202 may be based on the OpenMP standard. The compiler 26 may be a different compiler for different operating systems and/or different hardware. In some embodiments, the compiler 26 may generate object code 210 that may be executed on Intel® processors. Specification, p. 6, ln. 26 – p.7, ln. 5.

Linker 30 may receive object code 210 and various routines and functions from a run-time library 206 and link them together to generate executable code 208.

In some embodiments, the run-time library 206 may contain function subroutines that the linker may include to support "!$omp reduce" directives. Specification, p. 7, ln. 6 – p.7, ln. 11.

Referring to Figure 3, the translator 28 may receive a program unit(s) 301. In some embodiments, a "program unit" may be a collection of statements in a programming language that may be processed by a compiler or translator. The program unit(s) 301 may contain a reduction operation 303. In response to the reduction operation 303, the translator 28, in some embodiments, may translate the program unit(s) 301 into a call to a reduction routine 307. In addition, the translator 30 may translate the program unit(s) 301 into a call to a reduction routine that may reference a generated callback routine 305. Specification, p. 7, lns. 12 – 22.

The term "callback" is an arbitrary name to refer to the routine 305. Other references to the routine 305 may be utilized. The translation of program unit(s) 301 into the two routines 305 and 307 may be performed, in some embodiments, using a source-code-to-source-code translation. The source code may be Fortran 90, Fortran 95, C, C++, or other source code languages. However, routines 305 and 307 may also be intermediate code or other code. Specification, p. 7, ln. 23 – p.8, ln. 3.

The callback routine 305 may be a routine specific to the reduction to be performed. For example, the reduction may be an add, subtract, multiply, divide, trigonometric, bit manipulation, or other function. The callback routine encapsulates the reduction operation as will be described below.

The routine 307 may call a run-time library routine (not shown) that may utilize the callback routine 305 to, in part, perform a reduction operation. As part of the call to the run-time library routine, the routine 307 may reference the callback routine 305 Specification, p. 8, ln. 4 – p.8, ln. 14.

Referring to Figure 4, a program unit 401 includes a reduction operation. Program units 403 and 405 are examples, in some embodiments, of routines 307 and 305 respectively that may be translated in response to the reduction operation in the program unit 401. Specification, p. 8, ln. 14 – p. 10, ln. 17. An example of the reduction operation illustrated in the program unit 401 may have, in some embodiments, the following form:

```
!$omp reduce reduction (argument(s))
...
!$omp end reduce
```

The program unit 403 is an example of a translation of the program unit 401 in accordance with block 307 of Figure 3. The reduction routine call in the program unit 403, in some embodiments, may have the following form:

```
Reduction_routine(callback_routine, variable1, ...)
```

The program unit 405 is an example of a translation of the program unit 401 in accordance with block 305 of Figure 3. This program unit 405 may contain source code, or other code, to perform an algebraic function to compute, in part, the reduction operation. Specification, p. 8, ln. 15 – p.9, ln. 10. For example, in some embodiments, to implement an addition reduction, the program unit 405 may contain the equivalent of the following code instructions:

```
Callback_routine(a0, a1)
a0= a0 + a1
return
end
```

Where a0 and a1 may be variables that may be passed to the program unit 405. However, in other embodiments, in response to a reduction directive with a vector or array reduction argument, the program unit 405 may perform vector or array reductions. A vector or array reduction may, in some embodiments, be implemented, in part, by a 1 or more dimension loop nest that performs the vector or array reduction operations. Specification, p. 9, ln. 20 – 27.

Also, multiple reduction operations may be combined, in some embodiments, so that a single reduction routine call may be utilized and a single callback routine may contain the code to perform the multiple reductions. By performing multiple reduction operations, an increase in performance and scalability of reductions operations may be realized as the associated processing and synchronization overhead may be reduced relative to performing separate reduction operations. Specification, p. 10, ln. 1 – 9.

Additionally, in some embodiments, a reduction on objects may be achieved. The objects may be referenced by descriptors and the address of the descriptors may be passed through the reduction routine call and into the callback routine, in some embodiments. A descriptor may include an address of the start of an object and may include data describing the size, type or other attributes of the object. Specification, p. 10, ln. 10 – 17.

In some embodiments, source-code-to-source-code translators may provide, in part, source-code translations while run-time library routines my be implemented using low-level instruction sets to support reduction operations on an individual platform in order to optimize performance on that platform. Such a combination of source-code translations and run-time library implementations, may, in some embodiments, provide a cost effective solution to optimize reduction operations on a plurality of computing platforms. Specification, p. 16, ln. 13 – 22.

For example, in some embodiments, a run-time library routine, that may perform a logarithmic reduction, may be optimized for a particular computer platform to partition the reduction operation between a plurality of threads. As previously described, partitioning the reduction operation such that each parallel thread may act to reduce a unique portion of the variables and then combining the reductions made by each parallel thread may increase the efficiency of the reduction operation, in some embodiments. Specification, p. 16, ln. 23 – p. 17, ln. 4.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Each of the following grounds of rejection are presented for review. Claims 1, 3, 6-8, 10-15, 18, and 20-26 stand rejected as follows:

<u>35 U.S.C. §103 Rejections over Poulsen in view of Sundaresan</u>

A. Claims 1, 3, 6-8, 10-15, 18, 20, and 21 stand rejected under 35 U.S.C. §103 over U.S. Patent No. 5,812,852 (Poulsen) in view of U.S. Patent No. 5,937,194 (Sundaresan);

B. Claim 23 stands rejected under 35 U.S.C. §103 over Poulsen in view of Sundaresan;

C. Claim 26 stands rejected under 35 U.S.C. §103 over Poulsen in view of Sundaresan;

<u>35 U.S.C. §103 Rejections over Poulsen in view of Sundaresan and Hardwick</u>

D. Claim 24 stands rejected under 35 U.S.C. §103 over Poulsen in view of Sundaresan and further in view of U.S. Patent No. 6,212,617 (Hardwick).

E. Claims 22 and 25 stand rejected under 35 U.S.C. §103 over Poulsen in view of Sundaresan and further in view of Hardwick.

VII. ARGUMENT

Claims 1, 3, 6-8, 10-15, 18, 20, 21, 23 and 26 stand rejected under 35 U.S.C. §103(a) over U.S. Patent No. 5,812,852 (Poulsen) in view of U.S. Patent No. 5,937,194 (Sundaresan). In addition, claims 22, 24 and 25 stand rejected under §103(a) over Poulsen in view of Sundaresan in further view of U.S. Patent No. 6,212,617 (Hardwick). Applicants respectfully traverse these rejections and request that the rejections be reversed.

A. Claims 1, 3, 6-8, 10-15, 18, 20, and 21 Are Patentable Under 35 U.S.C. § 103(a) over Poulsen in view of Sundaresan

With regard to claim 1, neither reference teaches or suggests translating a first program unit into two different additional program units in the manner recited by claim 1. That is, claim 1 recites translating a first program unit into a second program unit and also translating that same first program unit into a third program unit, where the second and third program units are to perform different recited functions.

The impropriety of the Examiner's rejection is clear. Simple reference to FIG. 1 of the primary reference, Poulsen, indicates the fallacy of the Examiner's rejection. Specifically, the Examiner contends that Poulsen teaches translation of a single program unit into two different program units. If this is so, where does the block diagram of FIG. 1 of Poulsen (indicated to be the "overall structure of the present invention") anywhere show two such translations? It does not. Instead, Poulsen simply teaches a single translation, namely translation of a parallel computer program 100 and its translation by translation means 120 into a second parallel computer program 130. Plainly, there is no third program unit that is generated based on another translation of the first computer program 100. Thus, the Examiner's reliance on Poulsen for teaching translation of a first program unit into multiple different program units is clearly erroneous.

Instead, the program units contended to be the translated units in Poulsen are: (1) "library calls (i.e., "a second program unit"); and (2) privatizable storage object declarations (i.e., "a third program unit")." Final Office Action, p. 2.

As to the contented library calls, all that Poulsen teaches is that such library calls are used to initialize various structures. Poulsen, col. 9, lns. 2-19. That is, Poulsen nowhere teaches or suggests that such library calls can be instructions to partition a reduction operation between multiple threads. Nor do the library calls reference a third program unit, where the third program unit is also translated from the first program unit.

With regard to the contended third program unit, i.e., privatizable storage object declarations, these program statements are just that, declarations of storage objects, and are not an encapsulation of a reduction operation (as conceded by the Examiner). Nor do such declarations of Poulsen anywhere teach or suggest performing an algebraic operation on a set of variables. Instead, all the Examiner contends is that these privatizable storage object declarations encapsulate a global storage object. Such a global storage object however nowhere teaches or suggests either a reduction operation or performance of an algebraic operation.

As such, the Examiner seeks to combine Sundaresan with Poulsen. However, Sundaresan nowhere discusses program translation or any reason why its program would need to be translated in the manner taught by Poulsen. Simply put, there is no teaching or suggestion in the references in order to combine them to obtain the claimed subject matter. Rather, the Examiner has clearly engaged in improper hindsight-based reconstruction. *In re Kotzab*, 55 U.S.P.Q.2d 1313, 1316-17 (Fed. Cir. 2000). In this regard, the Examiner merely states that it would have been obvious to combine the references "for the purpose of improving the expressibility and maintainability of the parallel computer program." Final Office Action, p. 5. Such improvement is merely the purported advantage of Sundaresan, and is not any valid motivation to combine Sundaresan and Poulsen. *In re Lee*, 61 U.S.P.Q. 2d 1430, 1435 (Fed. Cir. 2001).

Nowhere can the Examiner point to any disclosure in either of Poulsen or Sundaresan that teaches or suggests the desirability of modifying Poulsen such that its thread privatizable storage objects are somehow transformed into reduction operations. This is especially so, as Poulsen nowhere teaches the use or desirability of reduction operations. Instead, Poulsen is directed to transformation of a program such that multiple threads can access global storage objects in a privatized manner. Poulsen, col. 4, ln. 62-col. 5, ln. 6. Further, Sundaresan nowhere teaches or suggests that its parallel program be modified for use with such privatizable storage objects. As such, the modification proposed by the Examiner would change the principle of operation of Poulsen, running afoul of the MPEP's proscription against such a modification. MPEP §2143.01.

Accordingly, for at least these reasons, claims 1, 3, 6-7, and 21 are patentable over 35 U.S.C. § 103 and the cited references. For at least the same reasons, claims 8, 10-14, 15, 18 and 20 are patentable over 35 U.S.C. § 103 and the cited references.

B. Claim 23 Is Patentable Under 35 U.S.C. § 103(a) over Poulsen in view of Sundaresan

Claim 23 concerns using a run-time library to implement the reduction operation. The Examiner relies on Poulsen in this regard. Final Office Action, p. 12. In contrast, Poulsen merely addresses libraries and the privatization of storage objects. Col. 9, ln. 62—Col. 10, ln. 9. Neither Poulsen nor Sundaresan address using a run-time library to implement a reduction operation. For at least this further reason, claim 23 is patentable over 35 U.S.C. § 103 and the cited references.

C. Claim 26 Is Patentable Under 35 U.S.C. § 103(a) over Poulsen in view of Sundaresan

Claim 26 concerns a third program unit to perform an algebraic operation using a library. Again, the Examiner relies on Poulsen in this regard. Final Office Action, p. 12. In contrast, as stated above regarding claim 23, Poulsen merely addresses libraries and the privatization of storage objects. Col. 9, ln. 62—Col. 10, ln. 9. Neither Poulsen nor Sundaresan address a third program unit to perform an algebraic operation using a library. For at least this further reason, claim 26 is patentable over 35 U.S.C. § 103 and the cited references.

D. Claims 24 Is Patentable Under 35 U.S.C. § 103(a) over Poulsen in view of Sundaresan and Hardwick

Claim 24 concerns a third program unit to perform a plurality of vector operations. The Examiner relies on Hardwick in this regard. Final Office Action, p. 13. However, Hardwick merely addresses multiple *types* of operations (e.g., scans, reductions)—not a *plurality of vector operations*. Col. 6, lns. 34-5. Hardwick, nor any other cited reference, teaches or suggests a third program unit to perform a plurality of vector operations. For at least this further reason, claim 24 is patentable over 35 U.S.C. § 103 and the cited references.

In addition, for at least the same reasons discussed above regarding claim 1 (Section VII(A)), the rejection of this claim under §103(a) over Poulsen in view of Sundaresan and Hardwick is also clearly erroneous. That is, as described above, there is no motivation to combine Sundaresan with Poulsen. Even less motivation exists for the combination with Hardwick. In this regard, Hardwick is directed to reducing inter-processor communications during parallel processing. However, Hardwick is nowhere concerned with program translation, such as that of Poulsen. Nor is there any teaching or suggestion that would motivate one interested in performing program translations to implement the reduced inter-processor communications of Hardwick, which still does not teach or suggest the recited vector operations. For at least this further reason, claim 24 is patentable over 35 U.S.C. § 103 and the cited references.

E. Claims 22 and 25 Are Patentable Under 35 U.S.C. § 103(a) over Poulsen in view of Sundaresan and Hardwick

For at least the same reasons discussed above regarding claim 1 (Section VII(A)) and claim 24 (Section VII(D)), the rejection of claims 22 and 25 under §103(a) over Poulsen in view of Sundaresan and Hardwick is also clearly erroneous. That is, as described above, there is no motivation to combine Sundaresan with Poulsen. Even less motivation exists for the combination with Hardwick because Hardwick is directed to reducing inter-processor communications during parallel processing. Again, Hardwick is not concerned with program translation, such as that of Poulsen. Nor is there any teaching or suggestion that would motivate one interested in performing program translations to implement the reduced inter-processor communications of Hardwick. For at least this further reason, claims 22 and 25 are patentable over 35 U.S.C. § 103 and the cited references.

Applicants respectfully request that each of the final rejections be reversed and that the claims subject to this Appeal be allowed to issue.

Respectfully submitted,

Date: September 29, 2006

[illegible]

E.E. "Jack" Richards, II
Registration No. 53,514
TROP, PRUNER & HU, P.C.
1616 S. Voss Road, Suite 750
Houston, Texas 77057-2631
(512) 418-9944 [Phone]
(713) 468-8883 [Fax]
Customer No.: 21906

VIII. CLAIMS APPENDIX

The claims on appeal are:

1. A method comprising:

receiving a first program unit in a parallel computing environment, the first program unit including a reduction operation associated with a set of variables;

translating the first program unit into a second program unit, the second program unit including a set of one or more instructions to partition the reduction operation between a plurality of threads including at least two threads and to reference a third program unit; and

translating the first program unit into the third program unit, the third program unit including a set of one or more instructions that encapsulate the reduction operation to perform an algebraic operation on the variables.

3. The method of claim 1 further comprising reducing the set of variables logarithmically.

6. The method of claim 1 further comprising associating the plurality of threads each with a unique portion of the set of variables.

7. The method of claim 6 further comprising combining, in part, the variables associated with the plurality of threads in a pair-wise reduction operation.

8. An apparatus comprising:

a memory including a shared memory location;

a translation unit coupled with the memory, the translation unit to translate a first program unit including a reduction operation associated with a set of at least two variables into a second program unit, the second program unit to partition the reduction operation between a plurality of threads including at least two threads and to reference a third program unit, the translation unit to also translate the first program unit into the third program unit, the third program unit to encapsulate the reduction operation to perform an algebraic operation on the variables;

a compiler unit coupled with the translation unit and the memory, the compiler unit to compile the second program unit and the third program unit; and

a linker unit coupled with the compiler unit and the memory, the linker unit to link the compiled second program unit and the compiled third program unit with a library.

10. The apparatus of claim 8 wherein the variables in the set of variables are each uniquely associated with the plurality of threads and the library includes instructions to combine, in part, the variables associated with the plurality of threads.

11. The apparatus of claim 10 wherein the library includes instructions to combine, in part, the variables in a pair-wise reduction.

12. The apparatus of claim 8 further comprising a set of one or more processors to host the plurality of threads, the plurality of threads to execute instructions associated with the second program unit.

13. The apparatus of claim 8 wherein the third program unit includes a callback routine and the callback routine is associated with instructions operative to perform the algebraic operation on at least two variables in the set of variables.

14. The apparatus of claim 13 wherein the library is operative to call the callback routine to perform, in part, a reduction on at least two variables in the set of variables.

15. A machine-readable medium that provides instructions, that when executed by a set of one or more processors, enable the set of processors to perform a method comprising:

receiving a first program unit in a parallel computing environment, the first program unit including a reduction operation associated with a set of variables;

translating the first program unit into a second program unit, the second program unit including a set of one or more instructions to partition the reduction operation between a plurality of threads including at least two threads and to reference a third program unit; and

translating the first program unit into [[a]] the third program unit, the third program unit including a set of one or more instructions that encapsulate the reduction operation to perform an algebraic operation on the variables.

18. The machine-readable medium of claim 15 wherein the method further comprises reducing the variables, in part, logarithmically.

20. The machine-readable medium of claim 15 wherein the instructions cause the second program unit to utilize, in part, the third program unit to perform the reduction operation on the set of variables.

21. The method of claim 1, further comprising performing a plurality of reduction operations in the third program unit.

22. The method of claim 1, further comprising performing a vector reduction operation in the third program unit via a N-dimension loop in the third program unit.

23. The method of claim 1, further comprising using a run-time library to implement the reduction operation.

24. The apparatus of claim 8, wherein the third program unit is to perform a plurality of vector operations.

25. The apparatus of claim 8, wherein the third program unit is to perform a vector reduction operation via a N-dimension loop.

26. The apparatus of claim 8, wherein the third program unit is to perform the algebraic operation using the library.

IX. EVIDENCE APPENDIX

There was no evidence submitted during prosecution.

X. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.